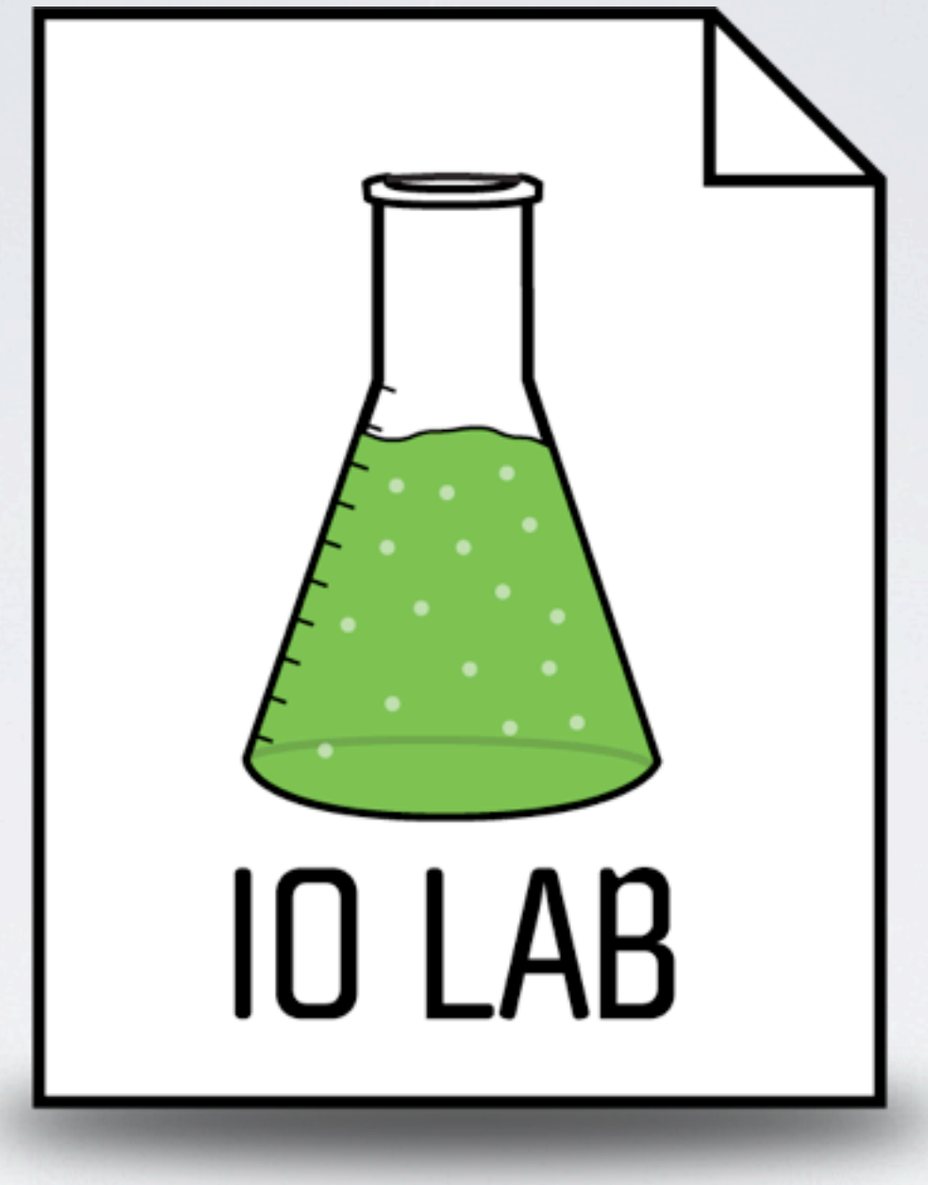


INFORMATION ORGANIZATION LAB

PROGRAMMING PARADIGMS

Imperative

Procedural

Object-Oriented

Declarative

Functional

This is a breakdown given by Wikipedia—read it and you'll known about as much as we do. One major distinction is that in imperative paradigms you track state changes whereas in declarative paradigms you connect inputs in outputs.

In procedural paradigms you describe step-by-step **how** to solve a problem. In a declarative paradigm you describe **what** the program should accomplish instead of how. **Regular expressions**, XSLT, and CSS are examples of declarative languages. Typical examples of functional languages include LISP, Scheme, and Erlang.

Many modern languages **mix paradigms**: in Python and JavaScript you can use procedural, object-oriented, and functional paradigms.

FUNCTIONAL PROGRAMMING

Map

Apply a function to every element in a collection.

Filter

Use a function to select elements in a collection.

Reduce

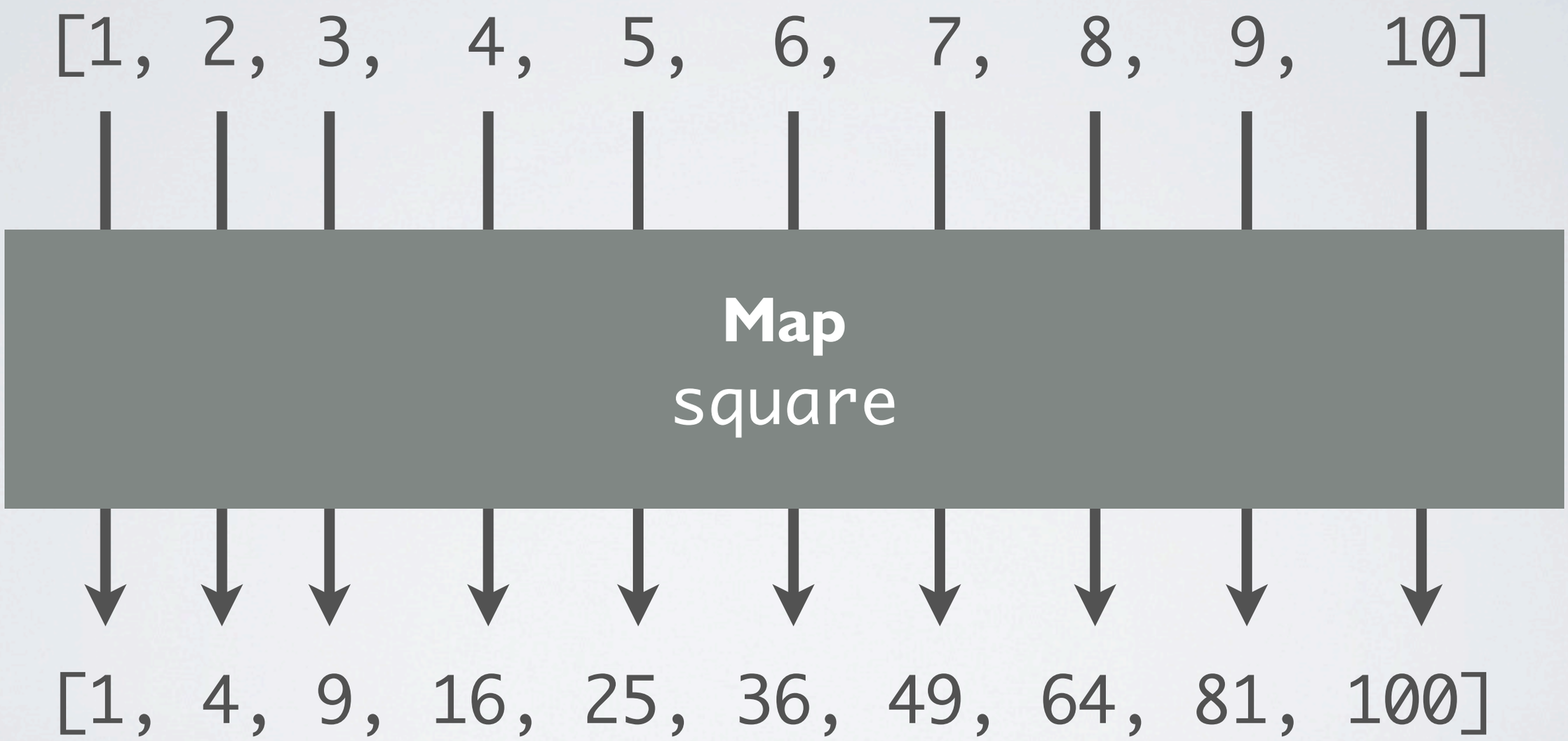
Use a function to combine all elements in a collection.

Someone from the CS department can give you a much better theoretical overview of functional programming and their advantages. For our more practical purposes, functional programming is effectively three techniques: map, filter, and reduce. We're applying this functional paradigm because it's a cleaner, shorter way of accomplishing some things we want to do in information retrieval.

Language support: **JavaScript** 1.6 introduced some functional methods like .map(), .filter(), .every(), .some(). These aren't supported in IE, but you can add them by extending the Array prototype. Code at https://developer.mozilla.org/En/Core_JavaScript_1.5_Reference/Objects/Array. jQuery also lets you use .map and .filter on jQuery objects.

Python has support for functional programming using methods like map, filter, reduce and list comprehensions.

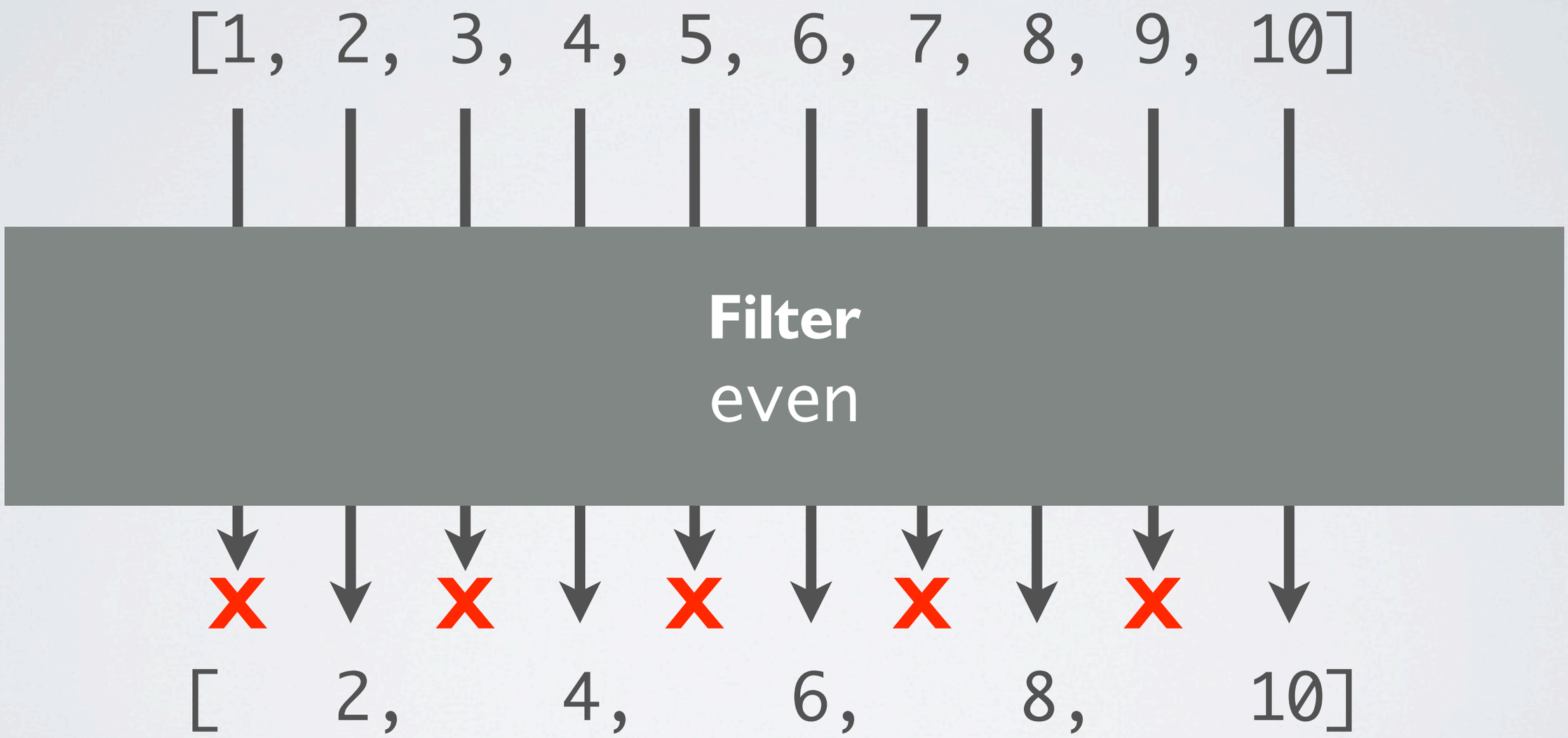
MAP FUNCTION



In the functional paradigm, you apply a function to elements in a collection. With a map operation, every element in the original collection gets mapped to an element in the new collection. The example here is a simple math operation—applying a square function to a list of numbers—but you can apply functions to any datatype you can think of, including strings and hashes.

Note that when you perform a map, you end up with the same number of elements that you started with.

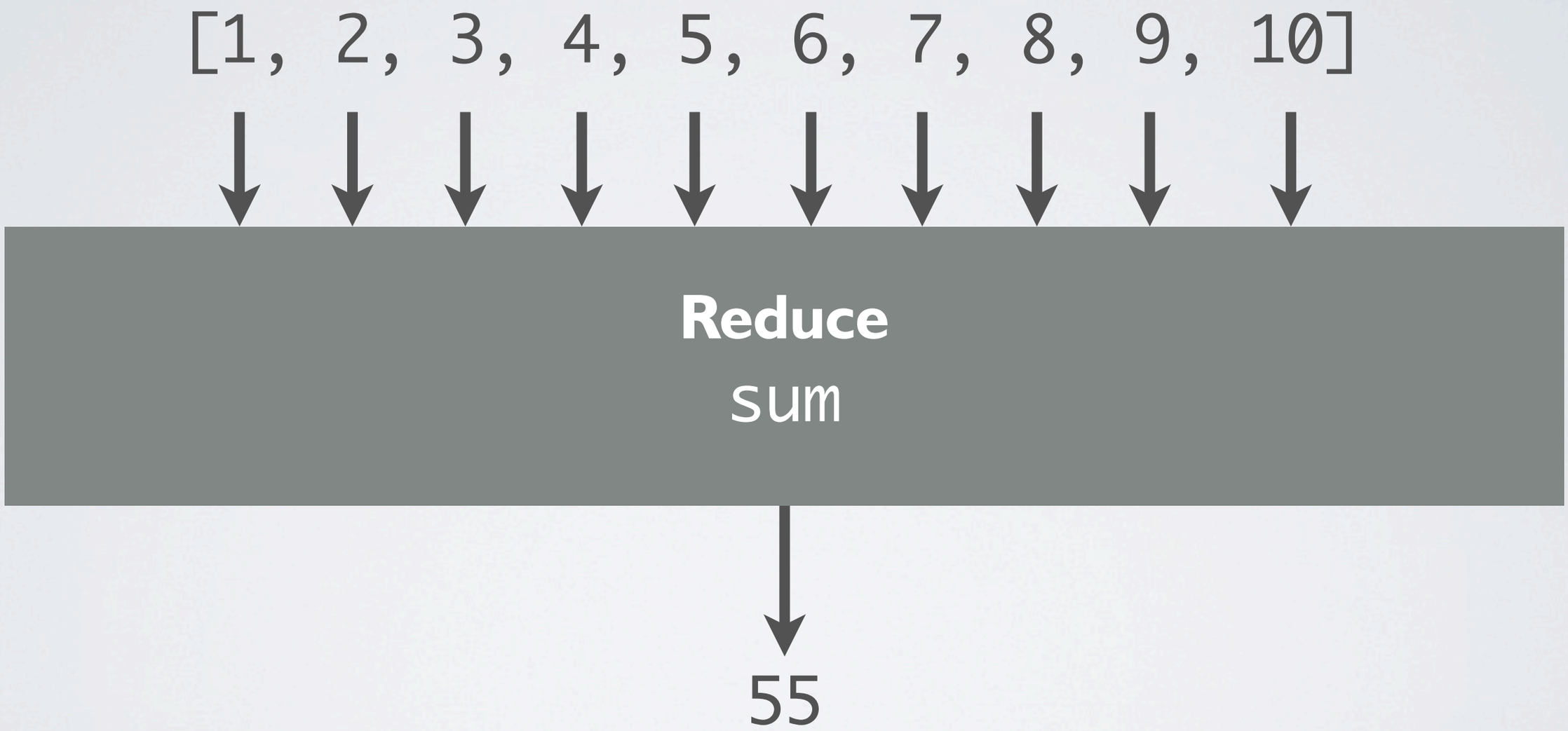
FILTER FUNCTION



With a filter operation you determine which elements will be in the new collection. The filtering function returns true or false, and those elements for which the function returns true are in the new collection.

In contrast with a map operation, there may not be the same number of elements in the resulting collection as the starting collection. The elements themselves, however, remain unchanged after the filter.

REDUCE FUNCTION



In a reduce operation, you combine all the elements in a collection one-by-one to a single value.

FUNCTIONAL JAVASCRIPT

Map

```
var nums = [1,2,3,4,5];

nums.map(function(n){
    return n * n;
})
```

Filter

```
var nums = [1,2,3,4,5];

nums.filter(function(n){
    if (n < 2 || n % 1 != 0) {
        return false;
    }
    for (var i=2; i < n; i++){
        if (n % i == 0) {
            return false;
        }
    }
    return true;
});
```

Reduce

```
var nums = [1,2,3,4,5];

nums.reduce(function(a,b){
    return a * b;
})
```

FUNCTIONAL JAVASCRIPT

```
var pixar = [
    {name: "Up", score: "98%", rating: "PG", year: 2009},
    {name: "WALL-E", score: "96%", rating: "G", year: 2008},
    {name: "Ratatouille", score: "96%", rating: "G", year: 2007},
    {name: "Cars", score: "75%", rating: "G", year: 2006},
    {name: "Finding Nemo", score: "98%", rating: "G", year: 2003},
    {name: "The Incredibles", score: "97%", rating: "PG", year: 2004},
    {name: "Monsters, Inc.", score: "95%", rating: "G", year: 2001},
    {name: "Toy Story 2", score: "100%", rating: "G", year: 1999},
    {name: "A Bug's Life", score: "91%", rating: "G", year: 1998},
    {name: "Toy Story", score: "100%", rating: "G", year: 1995}
];
```

```
pixar.map(
    function(movie){return movie.name}
);
```

```
["Up", "WALL-E", "Ratatouille", "Cars", "Finding Nemo", "The Incredibles",
"Monsters, Inc.", "Toy Story 2", "A Bug's Life", "Toy Story"]
```

You can use map to extract a simple set of information from more complex objects. Here we start with an array of objects, each with the name of a movie, its Rotten Tomatoes score, rating, and release year. After applying a map function that returns just the name of each movie, we have an array of strings.

FUNCTIONAL JAVASCRIPT

```
var pixar = [
    {name: "Up", score: "98%", rating: "PG", year: 2009},
    {name: "WALL-E", score: "96%", rating: "G", year: 2008},
    {name: "Ratatouille", score: "96%", rating: "G", year: 2007},
    {name: "Cars", score: "75%", rating: "G", year: 2006},
    {name: "Finding Nemo", score: "98%", rating: "G", year: 2003},
    {name: "The Incredibles", score: "97%", rating: "PG", year: 2004},
    {name: "Monsters, Inc.", score: "95%", rating: "G", year: 2001},
    {name: "Toy Story 2", score: "100%", rating: "G", year: 1999},
    {name: "A Bug's Life", score: "91%", rating: "G", year: 1998},
    {name: "Toy Story", score: "100%", rating: "G", year: 1995}
];
```

```
pixar.filter(
    function(movie){return movie.rating == "PG"}
);
```

```
[{name: "Up", score: "98%", rating: "PG", year: 2009},
{name: "The Incredibles", score: "97%", rating: "PG", year: 2004}];
```

We can similarly use filter to determine which movies received a “PG” rating. In this case, what we get back is an array of objects. These objects look the same as the objects in the original array.

If we wanted just the names of the movies rated PG, we could combine the filter shown here with the map shown on the previous slide.

FUNCTIONAL PYTHON

Map

```
def square(x):
    return x * x

map(square, range(10))
```

Filter

```
def prime(x):
    if x < 2 or x % 1 != 0:
        return False
    for y in range(2, x):
        if x % y == 0:
            return False
    return True

filter(prime, range(100))
```

Reduce

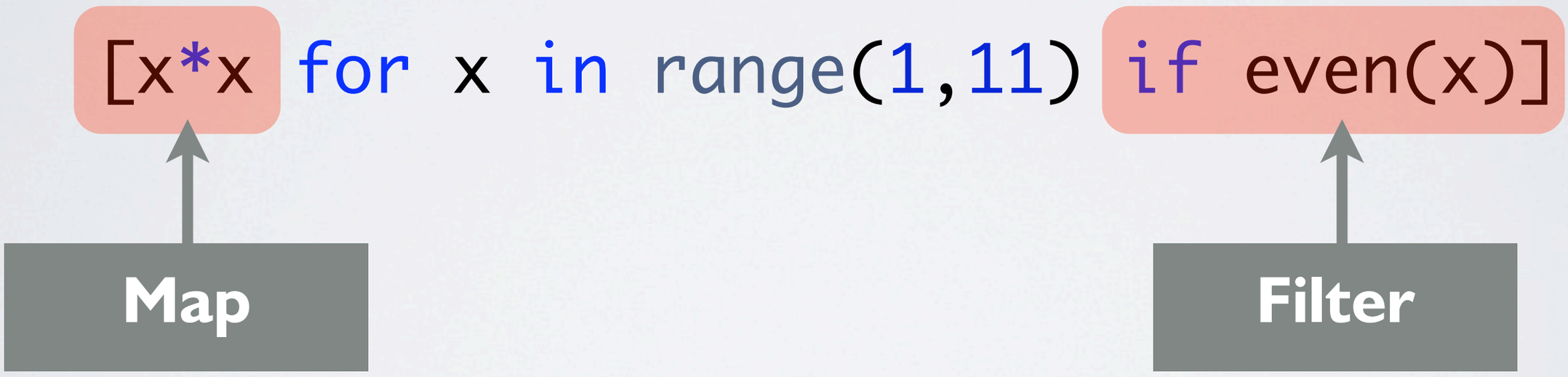
```
def product(a,b):
    return a * b

reduce(product, range(1,5))
```

Python has map, filter, and reduce methods which work much like the similarly-named methods in JavaScript. They accept a function and a collection to apply that function to. See http://docs.python.org/tutorial/datastructures.html#functional-programming-tools.

You can define functions inline using the lambda syntax. http://docs.python.org/reference/expressions.html#lambda.

FUNCTIONAL PYTHON

List Comprehensions



More often in Python you see the functional pattern used in list comprehensions.
See http://docs.python.org/tutorial/datastructures.html#list-comprehensions for more information.

TWITTIR

This is a framework for loading a corpus of messages and applying different search algorithms to compare their effectiveness. See demo on course website.

Together in class we implemented a simple string search and made it case-insensitive. Then we used a JavaScript version of the Porter Stemmer to improve recall by treating different tokens (like “learn” and “learning”) as the same.

Improvements to search algorithms include doing a term or boolean search instead of a simple string search, devising a ranking system for results (using tf/idf), or searching using the metadata provided for each document.

80legs

80Legs is a service that provides you with access to a distributed spidering network to process document collections. In today's demonstration we're going to write a program to do the analysis on fetched pages. 80Legs takes care of most of the crawling details for you, like throttling, obeying robots.txt, duplicate URLs, link extraction, and so forth. Also: it's Java.

80legs requires you to work within a strict set of limitations in order to provide scaling. You have no common state between instances of your program, and there is no way to communicate with the file system, the internet, anything.

Our example is looking at the usage of the rel="license" microformat on the Internet, specifically the selection of different jurisdictions for CC licenses. See sample code on course website.

80 LEGS

```
$("a").map(function(){
    return processDocument(this);
}).get().join(",");
```

MAP/REDUCE OF THE WEB

PROJECT 5

- Create “drops” to collaborate where you can chat, share files, have a conference call, receive voicemail.

FOR NEXT WEEK

- Start thinking about project ideas and groups for project 5.